ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)

https://doi.org/10.17559/TV-20190105200327
Subject review

Multi-Criteria Decision-Making Approach for Container-based Cloud Applications: The SWITCH and ENTICE Workbenches

Polona ŠTEFANIČ, Vlado STANKOVSKI

Abstract: Many emerging smart applications rely on the Internet of Things (IoT) to provide solutions to time-critical problems. When building such applications, a software engineer must address multiple Non-Functional Requirements (NFRs), including requirements for fast response time, low communication latency, high throughput, high energy efficiency, low operational cost and similar. Existing modern container-based software engineering approaches promise to improve the software lifecycle; however, they fail short of tools and mechanisms for NFRs management and optimisation. Our work addresses this problem with a new decision-making approach based on a Pareto Multi-Criteria optimisation. By using different instance configurations on various geo-locations, we demonstrate the suitability of our method, which narrows the search space to only optimal instances for the deployment of the containerised microservice.This solution is included in two advanced software engineering environments, the SWITCH workbench, which includes an Interactive Development Environment (IDE) and the ENTICE Virtual Machine and container images portal. The developed approach is particularly useful when building, deploying and orchestrating IoT applications across multiple computing tiers, from Edge-Cloudlet to Fog-Cloud data centres.

Keywords: Cloud; Edge; Fog; Internet of Things; Non-Functional Requirements; Software Engineering; Quality of Service

1 INTRODUCTION

With the emergence of the Internet of Things (IoT), a great variety of new smart applications have become possible. New IoT applications are being built for homes, cities, communities, driving, industry, health, and so on. When developing such applications, software engineers have to address a plethora of Non-Functional Requirements (NFRs). Some IoT devices generate data streams at the Internet Protocol gateways, which have to be communicated across the Internet with very low latency, high throughput, low packet loss and low jitter. Devices such as smartphones, air-quality sensors, signalisation, cars and robots usually run on batteries for which it is necessary to achieve high energy efficiency. Another type of applications requires the storing and processing of data in real time and response to various events in real time as well. Being able to balance various NFRs properly, for example, response time versus energy consumption and operational cost, represents a critical factor for the success of software companies active in the area.

Modern software engineering environments for Cloud computing support graphic composition and management of containers usually designed as microservices and packed in containers and Virtual Machine images [1]. The full technology stack includes Operating Systems optimized for containers (e.g. RancherOS, CoreOS), container management middleware (e.g. Docker, Singularity), overlay networking and orchestration (e.g. Swarm, Kubernetes) and Interactive Development Environments (e.g. Juju [2], Fabric8 [3]). Using these modern technologies, it is already possible to radically improve the software lifecycle. However, these technologies fail short of tools and methods for NFRs management and balancing, and they do not take full advantage of emerging Fog and Edge Computing concepts and approaches [4, 5]. By using the same technology stack, it is now possible to build and orchestrate container-based applications across multiple computing tiers from Cloud data centres to the Fog and Edge of the network.

Since every IoT application's deployment scenario is unique (e.g. different number of sensors, robots, cars and cameras, different available infrastructures and business ideas), the management and balancing of NFRs require a great deal of delicacy (see Fig. 1). Some network-intensive containers are best placed at computing infrastructures close to the IP gateways [6]. Many companies active in the telecommunications domain already provide new Fog Computing systems, such as microservers and routers that can be used both for offloading computations from the IoT devices and as Big Data gatekeepers for the data centres. IoT applications are thus destined to run across multiple tiers, that is, multiple administrative domains, public and private Clouds. In many situations, the IoT devices may dynamically change geographic locations, so it is necessary that the application orchestration process maintains high Quality of Service (QoS) dynamically, while moving containers along the entire Cloud paradigm from data centers to the Fog, or from one Edge location to another [7].

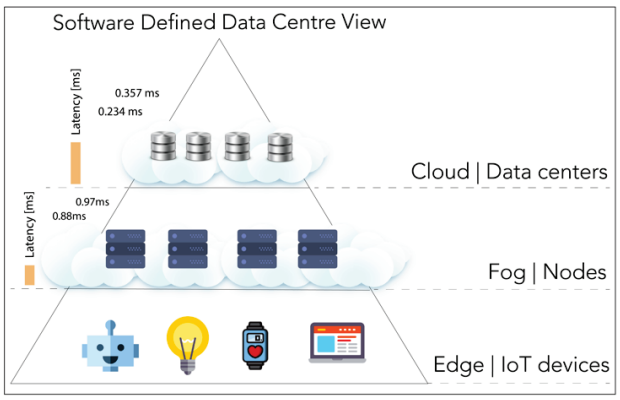

Figure 1 Deploying containers across multiple computing tiers for high QoS

2 RELATED WORK

New tools for software engineering are gaining popularity allowing quick development of cloud-native applications. For example, Juju [8] is an open source component-based graphical modelling tool for service-oriented architectures and application deployments with a set of predefined containers connected (e.g. charms). An

application developed in Juju can be scaled vertically and horizontally, thus benefitting from some critical properties of the Cloud. Fabric8 [9] is another open source platform with a console for creating, building and deploying microservices, as well as running and continuously managing them.

Emerging software engineering tools have also gained the attention of software engineering researchers. Particular research interest is concentrated on NFRs management and deployment scenarios on cloud-to-edge infrastructures. New deterministic approaches, such as those of Garg et al. [10] and Karim et al. [11] are based on the Analytic Hierarchy Process (AHP), which considers QoS requirements for ranking cloud deployment options. The drawback of these methods is the computational complexity, which renders such methods not suitable in existing software engineering practices.

Several studies address complex VM placement and service allocation issues by utilising bin packaging algorithms. Wang et al. [12] use stochastic bin packing algorithm, which addresses CPU and bandwidth related requirements. Srikantaiah et al. [13] defineVM consolidation problem as a multi-objective bin packing problem and consider trade-offs for minimisation of resource utilisation and energy consumption and maximizing the performance of consolidated workloads.

The VCRA-CP approach by Zhang et al. [14] meets the desired user's QoS requirements by reducing the cost or resource usage during resource provisioning. CPU, memory, bandwidth and network latency are the NFRs that Singh et al. [15] use in their VectorDoc algorithm. Its strength is in dynamic load balancing and managing overloaded nodes. Ferdaus et al.[16] model the problem of VM placement as an NP-Hard Multi-Dimensional Vector Packing Problem (mDVPP) focusing on balancing the cloud resource utilisation as power consumption and resource wastage by proposing the use of AntColony Optimisation (ACO) meta-heuristic. All approaches consider NFRs as equal for the optimisation process, which is computationally complex and time-consuming.

Table 1 NFR management methods and their integration within modern software engineering workbenches

Study	NFRs considered	Virtualisation	Method	Addressed Problem	Integration
[10]	CPU, cost, memory, availability	Virtual Machines	Deterministic AHP	Cloud service selection ranking	SMI Cloud Framework
[11]	QoS requirements	Virtual Machines	Deterministic AHP	Cloud service selection ranking	No
[12]	Network bandwidth, number of servers	Virtual Machines	Stochastic bin packing algorithm	VM consolidation	No
[13]	CPU	Virtual Machines	Bin packing algorithm	Energy efficiency in data centers	No
[14]	Compute resources (CPU, memory)	Virtual Machines	Multi-objective bin packing	Minimising resource utilisation, energy consumption	No
[15]	Resource constraints	Vdisks, Virtual Machines	VectorDot	Load balancing in data centers	No
[16]	CPU, memory, network I/O	Virtual Machines	ACO metaheuristics	Minimising power consumption, resource wastage	No
[17]	Computing time, user requests, power	Virtual, Physical Machines	GABA	Conserving power consumption	No
[18]	Cost, bandwidth, workflow makespan	Virtual Machines	MOHEFT	Makespan, high-performance and economic cost	No
[19]	CPU, memory, storage, latency	Virtual, Physical Machines	Two-stage MO heuristic algorithm	Reduce energy consumption, network performance	No
[20]	QoS parameters, latency, throughput	Virtual Machines	ROAR	Horizontal scaling at low possible price	No
Present approach	Multi-level hard and soft QoS parameters	Virtual Machines and Docker containers	Multi-criteria bin packing approach with scalability	Edge, Fog and Cloud computing; Cloud-native time-critical applications engineering	Yes; SWITCH and ENTICE workbenches

Genetic multi-objective algorithms have been used for multi-criteria optimisation. Mi et al. [17] propose a Genetic Algorithm Based Approach (GABA) for adaptive self-reconfiguration of VM reallocation in large-scale data centres. GABA deals with multi-objective optimisation and returns the optimal number of VMs for each application whereby it takes into account physical locations according to time-varying requirements and dynamic environmental conditions. Durillo et al. [18] extend Heterogeneous Earliest Finish Time (HEFT) into Multi-Objective HEFT (MOHEFT) to return a set of non-dominant trade-off solutions among monetary cost and makespan of the workflow. According to their evaluation, MOHEFT returns better solutions than SPEA2 genetic algorithm. However, the communication cost is only used in the task ranking phase and not addressed in the infrastructure planning phase. Those approaches consider only a few pre-defined NFRs or they put all NFRs as an input into the algorithms, which makes the approach due to large number of parameters computationally very complex and not suitable for integration into software workbenches, such as those already mentioned. Dong et al. [19] have, therefore, designed a two-stage heuristic algorithm for the NP-hard problem for creating trade-offs among energy and performance regarding VM consolidation. In the first stage the number of physical machines and network elements is

used to reduce the energy consumption in data centres. In the second stage the optimisation of network performance is considered by migrating VMs.

In the ROAR framework [20], the VMs are scaled horizontally until the required QoS is achieved at the lowest possible price. ROAR aims to optimize cloud resource allocation decisions to meet QoS goals for web applications. ROAR describes the configuration of the web application, and the expected QoS requirements (e.g. throughput and latency). It works well in cloud environments where there are few available machines.

Tab. 1 compares related works with the present study. State-of-the-art approaches mostly consider only a few QoS parameters with the intention to reduce the computational complexity that would otherwise arise if all QoS attributes are fed as input to an optimisation algorithm. However, due to dynamically changing network conditions and the correlated nature of QoS parameters, considering trade-offs among only few QoS parameters may not always lead to expected results. For the decision-making process, it is necessary to dynamically consider QoS parameters for each application individually, which would make the approach useful for integration with workbenches such as SWITCH. The present approach contributes to the state-of-the-art by allowing a distinction between hard and soft QoS parameters. On the basis of mandatory/ hard QoS parameters (CPU, memory utilisation) which are necessary for particular service to be properly provisioned it eliminates inappropriate infrastructures (e.g. VMs) and in a specially designed multi-criteria decision making process considers only soft (desirable) constraints which have significant influence on the application's QoS. This makes it possible to dynamically narrow down the search space and shorten the optimisation time.

3 STUDY GOALS

The goal of our research was to enhance the software engineering process, that is, the DevOps lifecycle for smart IoT applications with an innovative NFR management and decision-making solution. Our practical goal was to implement the developed process by designing, developing and integrating it into two workbenches, the SWITCH Interactive Development Environment [21] and the ENTICE Virtual Machines (VMs) and container images portal [22], which both require a multi-criteria optimisation process. Our new and innovative Multi-Criteria Decision-Making (MCDM) approach for NFRs trade-off management underpins these two advanced workbenches [4].

4 MOTIVATING IOT APPLICATIONS

Several emerging smart IoT applications motivate this work. Following are two of them.

The company BEIA Consult[1] from Romania develops an IoT-based disaster early warning system. The NFRs investigated include response time, reliability and secure system operation. The application is built from sensors, and on-site signalisation deployed geographically in addition to various container-based software services developed to run in a Cloud. The sensors stream measurements to an Internet Protocol (IP) Gateway, for example, the TA900eor the Cisco-ASA, and from there to the software services running in the data centre. Developed containers include a time series database, based on Apache Cassandra, an automated emergency centre services, such as e-mail, short text messages (e.g. push notifications), voice call via Internet telephony and other notification services, that are developed to run on Apache Web server. The company Wellness Telecom[2] from Spain develops a Cloud based collaborative real-time communication application, which is business critical in the sense that it must operate without disruptions for its users in cases of greatly varying bandwidth and other conditions on the Internet. The NFRs are to reduce operational costs for the applications while maintaining high QoS. Developed containers include audio and video communications services, such as Voice over Internet Protocol (VoIP) servers, Multipoint Control Unit (MCU) Media mixer, instant messaging services and other.

The company MOG Technologies[3] from Portugal develops a Cloud application for live event broadcasting, which is time critical. NFRs investigated include Peak Signal-to-Noise Ratio (PSNR) and a variety of QoS metrics, including jitter, latency, packet loss, bandwidth and throughput. High Definition (HD) or H.264 video streams from a variety of devices, such as cameras and smartphones are fed to container-based services. The latter include encoders, decoders, format transformers, a video stream selector, temporary high-speed video files storage for semantic search and reuse and others.

Applications like these have to address multiple NFRs during their deployment from the Edge to the Cloud. A container-orchestration component that can operate across multiple computing tiers may provide more appropriate networking, computing, memory, storage and other resources, tailored to individual containers and microservices. Software engineering process for such applications should be supported by modern tools, including SWITCH and ENTICE.

4.1 Use Cases and Proof-Of-Concept Application

The first use case to which the new MCDM approach is applied is the container image delivery time. Executable images can have significant size (200 MB and more) and are usually stored in repositories. High availability of executable images can be achieved by replicating them in repositories geographically, which in turn multiplies the storage costs. This requires balancing various NFRs.

The second use case concerns the actually achieved QoS when running the microservice. For example, it is necessary to achieve high throughput between the Edge device and the user's smartphone. The selection of the Edge device should therefore depend on the availability of high throughput between the client and the HTTPS server.

The two generic use case scenarios are illustrated in Fig. 2. Every single time a specific functionality is needed, a container image is delivered, deployed and used in a

[1]http://www.beiaro.eu/
[2]http://www.wtelecom.es/welcome/en/
[3]https://www.mog-technologies.com/

selected Fog or Edge computing device. Once the required functionality is served, the container instance is terminated.

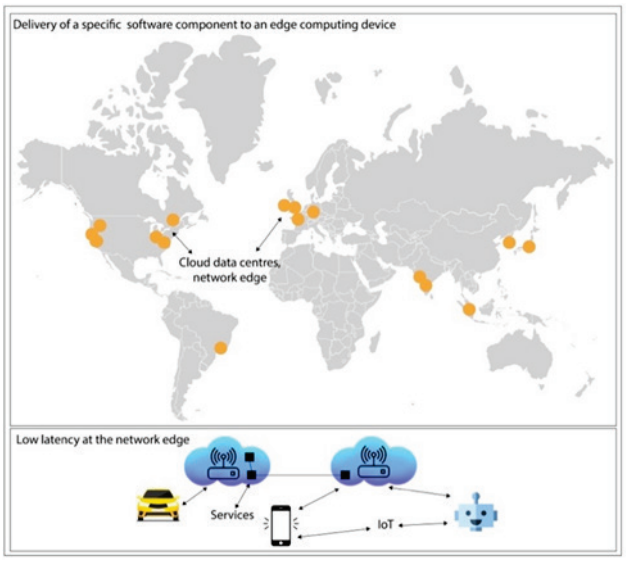

Figure 2 Use cases: (1) event-driven high-speed delivery of container images to Edge devices and (2) low-latency, high QoS of container-based services at the network Edge

For illustration a "Hello World" type of application was developed. It can be used by a person to take a photo or video and upload it to a Cloud storage. The file uploading functionality is implemented in a container with an open source Secure HyperText Transfer Protocol (HTTPS) server. The Web server runs continuously in the data center, while the container-based HTTPS server is designed to start in proximity of the client at Fog and Edge computing devices. In order to serve a file upload event, it is first necessary to deliver the container to the designated Edge device in proximity of the client. An orchestrator launches the HTTPS server container within a second, and then the file is uploaded. Once the process finishes, the HTTPS server transmits the uploaded file to a persistent storage service, which the user has established at an Amazon EC2 compliant storage. In this way, our application is innovatively designed to run dynamically, globally, and across multiple computing tiers. The logic tier (containing the HTTPS server) is developed and provided in a storage repository as a container image.

5 THE SWITCH AND ENTICE WORKBENCHES

In the course of our work, the authors designed and developed two innovative workbenches, namely, the SWITCH IDE [23] and the ENTICE VM and container images portal [22]. The new MCDM approach is used by both workbenches.

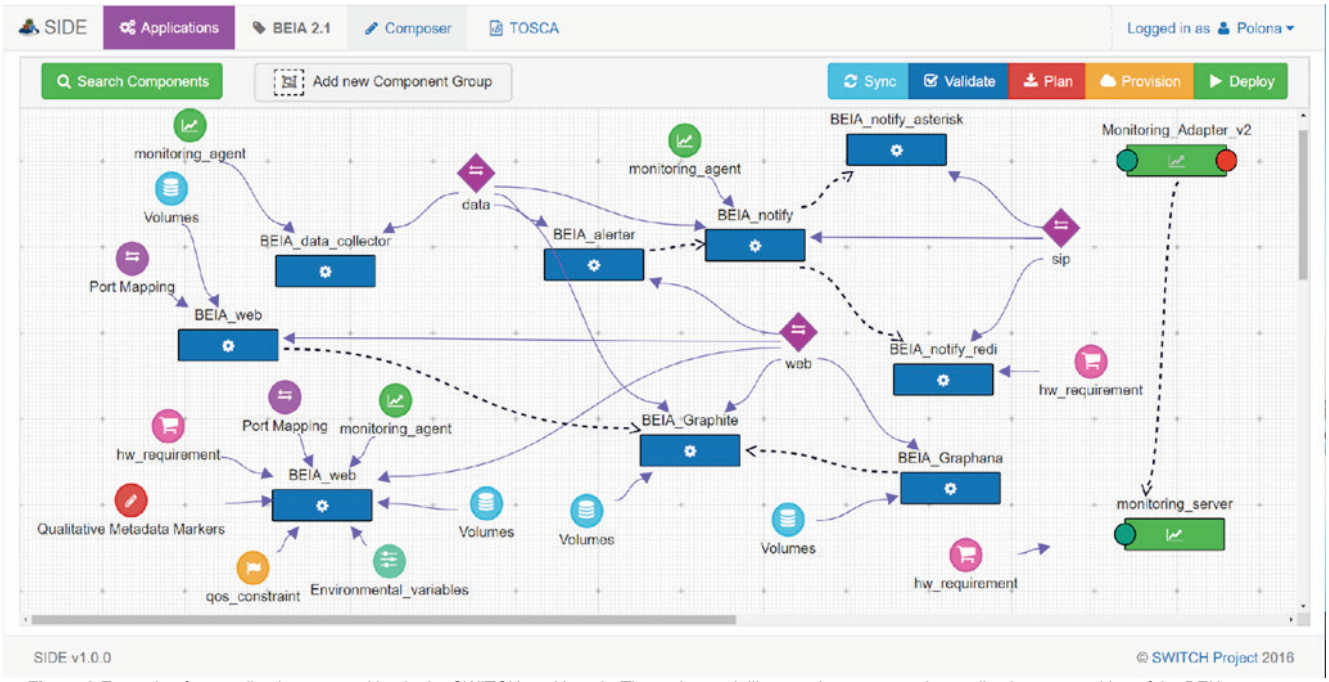

Figure 3 Example of an application composition in the SWITCH workbench. The entire modelling graph represents the application composition of the BEIA use case

The SWITCH IDE allows for component creation, application logic and workflow composition, container customisation, NFR management and decision-making, planning and provisioning strategies, monitoring and application deployment to a Cloud environment. The software engineering process starts by searching and fetching Docker images from public repositories, such as DockerHub. The software engineer puts containers on the SWITCH IDE canvas (dark blue containers, see Fig. 4) and customizes them by linking to them various properties (coloured circles) [24]. Additionally, monitoring system that contains monitoring probes, server and agents can be attached to the component as well. The containerised components can be linked to one another. In the current version of the SWITCH IDE the following properties can be set: hardware requirements, host related: CPU clock frequency, number of CPU cores, disk size, memory size.

All configured information represents basis for the management and balancing of NFRs, which is introduced by the new MCDM approach. The decisions taken by the software engineer are stored in a fully specified TOSCA specification (expressed in YAML). This file also includes the setup of public IP address, ports, domains and similar.

Following this step, the newly developed container-based application is ready for the deployment in Fog or Edge infrastructures.

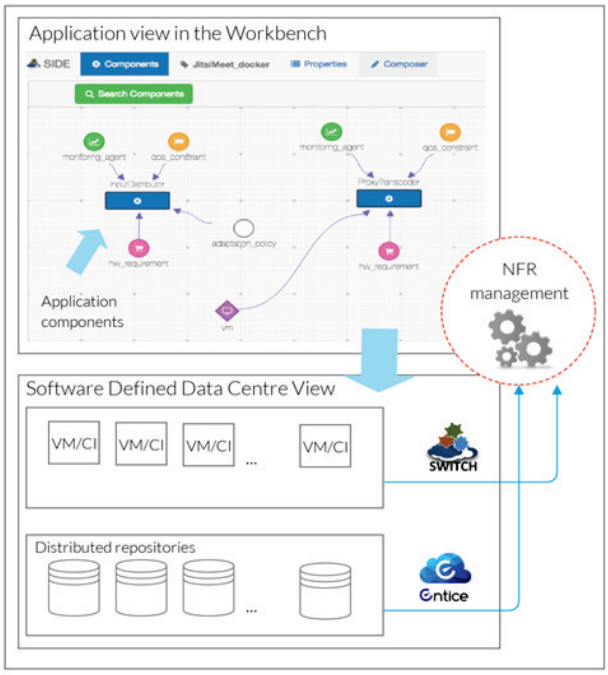

Figure 4 NFR management and balancing for the SWITCH and ENTICE workbenches

5.1 Underlying Services

The two workbenches that act as standalone services rely on a QoS Monitoring System, a Dynamic Real-Time Infrastructure Planner (DRIP) [26] and an Autonomous System Adaptation Platform (ASAP) [27]. The QoS monitoring system was implemented by using JCatascopia [28] as part of the SWITCH project. It allows for monitoring a great variety of metrics at multiple levels, including infrastructure, network, container, VM and application levels. The system can collect metrics:

- before deployment of the container image,
- during container runtime and
- before termination of the service.

The monitoring data is stored in a time series database based on Apache Cassandra. This database is used extensively for the NFR management and balancing based on a MCDM module. A Dynamic Real-time Infrastructure Planner (DRIP) automates the planning and provisioning a customized virtual Cloud infrastructure based on QoS constraints of the application. DRIP also allows deployment of application containers and their execution on demand. An Autonomous System Adaptation Platform (ASAP) takes monitoring data and autonomously controls the application in its runtime environment in order to maintain optimal QoS of the entire application. It provides a Global Cluster Manager developed on top of Docker and Kubernetes.

5.2 Multi-Criteria Decision-Making Approach

In order to address the requirements of emerging smart IoT-based applications, a new Multi-Criteria Decision-Making (MCDM) approach was designed and implemented. It aims at supporting the software engineer's decision-making process.

The realisation of the two above defined generic use cases depends on the ability to balance their NFRs properly. Fig. 5 illustrates the entire process. First, the software engineer requires some tools helping her/him to organize containers into computing tiers. The multitier application arrangement is represented in the SWITCH graphical Interactive Development Environment (IDE). In the next step of the process, the software engineer is able to specify various QoS requirements for particular containers (such as the throughput for the HTTPS server). In the third step of this process, environmental conditions for smooth operation of the service can be specified. Then, based on a variety of preliminary collected QoS monitoring metrics and Service Level Agreements (SLAs), it is possible to use the collected data for the analysis.

The result of the MCDM analysis, which is shown to the software engineer in the workbenches, takes the form of a Pareto front non-dominant solutions allowing the engineer to study the NFRs trade-offs. From the Pareto front the engineer can select one point from the chart - a point, which is lying on the Pareto front, thus specifying optimal deployment condition (e.g. VM in the cloud data center or edge node) for the container. Once this is done, in the fifth step of the process, the application logic with decision about the selected trade-off among conflicting NFRs is mapped into TOSCA.

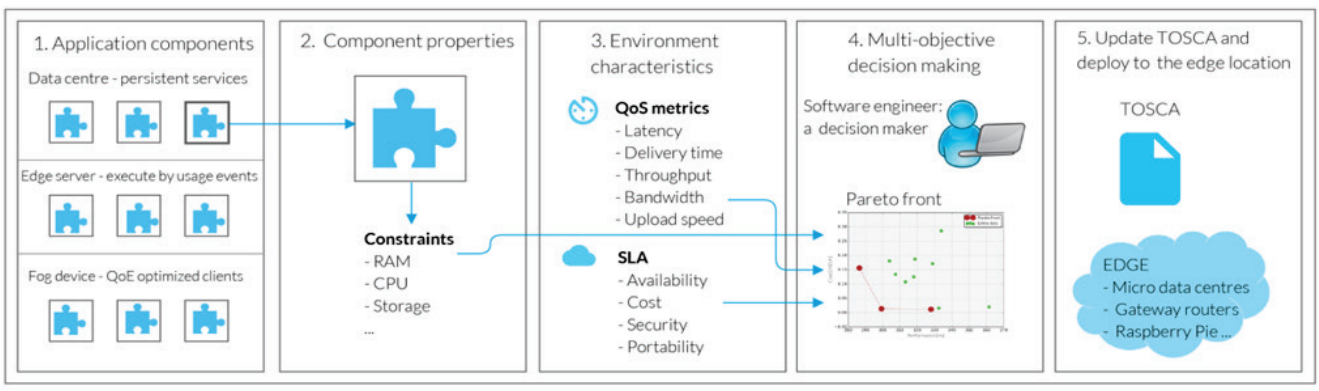

Figure 5 NFR balancing and decision making process for container-based services

Following this, the container images are deployed in one or more repositories along edge-cloudlet/fog-cloud data centres and their Uniform Resource Identifiers (URIs) are registered within the SWITCH IDE for further use

This new MCDM approach was implemented by using the Open Source JMetal Java framework [29] with focus on scalability of the method for multiple NFRs. It takes as an input monitored QoS metrics, SLA information of the various Cloud infrastructure providers and container's NFRs and hardware constraints. As a result, the algorithm returns a Pareto front [30], which presents a set of non-dominant solutions. A solution x dominates over x' (x, $x' \in X$) if x' is greater than x in a relation to all objective functions and x' has worse value for at least one of them. The solution x' presents a non-dominant solution, if there are no further solutions x that would dominate over x'. The set of solution $P' \in P$ represents a Pareto front [31].

6 RESULTS

The introduced MCDM part in the DevOps process is evaluated by using the developed proof-of-concept application which serves the already described two generic use cases. The Amazon Web Services Elastic Compute Cloud (AWS EC2) is used for benchmark testing. Amazon's EC2 currently offers VMs with different computing resources in data centres allocated in 21 regions worldwide. For the operation, the authors could use various VM instances with different performance and computing specifications, such as t2.micro, m4.large and m3.2xlarge. Although EC2 was used for benchmark testing, the solution is independent of the chosen Cloud provider.

The developed Global Cluster Manager (GCM) was used to orchestrate the HTTPS container image across the regions. The QoS monitoring system measured the following metrics for the application:

- Photo (file) upload time / ms. It shows the time it takes to transfer one given file to an HTTPS server deployed in a specific Edge or Fog device.
- Load time / ms. Represents the time needed for the HTTPS server to start.
- Upload speed / Mbit/s. It represents the speed with which the container image is transferred to the target Edge or Fog device.

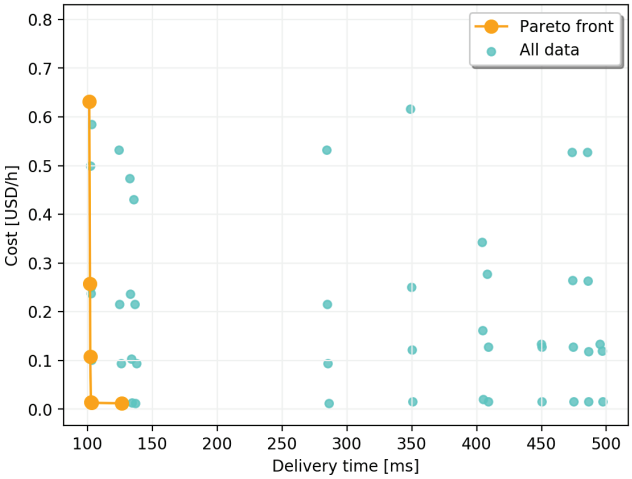

Figure 6 Container image delivery time

QoS metrics from this benchmarking process are fed into the MCDM algorithm, which returns Pareto front non-dominant solutions for the two investigated use cases. Fig. 6 concentrates on the container image delivery time for this service in a relationship to the cost of a specific VM, which depends on the images repository location. Obviously, if the VM is geographically far away the image delivery time is prolonged, which influences the QoS.

Fig. 7 shows the photo upload time versus cost. Green points represent all VMs (e.g. on edge devices), while the red points represent only the Pareto front non-dominant solutions. Obviously, the photo (or indeed any file) upload time significantly depends on the placement of the HTTPS server in the Edge-Cloudlet/Fog-Cloud continuum.

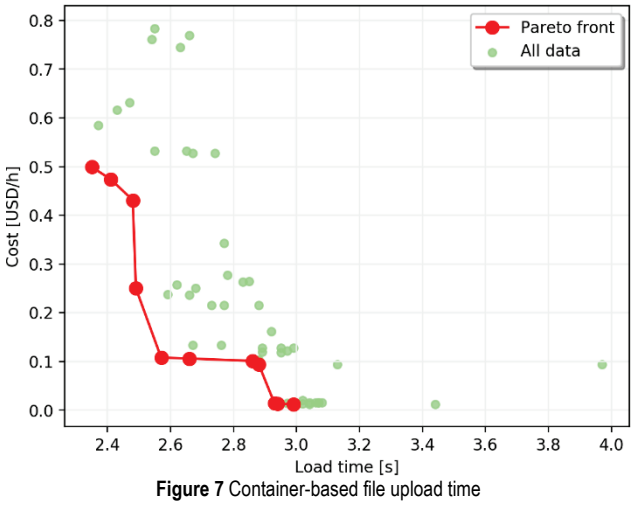

Figure 7 Container-based file upload time

By using these charts, the software engineer can fine tune its containers and microservices for execution. The engineer's decision relies on the trade-offs which can be studied from the chart. The optimal decision is stored in a TOSCA specification. This helps the software engineer to avoid many suboptimal solutions, which provide for neither low operational cost nor high QoS and automate the process. The generated graphs show many suboptimal choices that can be effectively avoided by using the newly designed and implemented MCDM approach.

7 DISCUSSION

With the emergence of IoT-based applications for smart cities, homes, communities, driving, industrial automation and so on, providing QoS to applications has become one of the major goals in software engineering. However, the QoS and NFRs can be many and conflicting. This work addresses these challenges by developing an approach and scalable method for the balancing and management of NFRs along the software lifecycle. The use of TOSCA specifications in the runtime environment is for interoperability purposes.

Many smart applications have a time-critical component, which can be addressed by moving computations to the Fog and the Edge of the network. In this context, complex container-based applications can be optimized across multiple tiers, which is an open research problem. Various elasticity mechanisms can be imagined, such as offloading computations from a robot to Fog devices with the intention to achieve greater energy

efficiency or to elastically scale the application in case of increasing workload. All this requires QoS and NFR balancing.

The present work contributes to the activities of the Software Engineering for Services and Applications [32] cluster of Horizon 2020 projects.The newly launched DECENTER[4] EU-Korean Horizon 2020 project currently plans to use the results of this study. The DECENTER project implements a Fog Computing Platform that will provide highly distributed Artificial Intelligence methods at the Edge and Fog. When designing highly computationally and network intensive applications, such as AI applications, it may be appropriate to rely on scalable algorithms and approaches suitable for Fog and Edge computing, such as the MCDM approach presented in this study.

8 CONCLUSIONS

In this work an innovative Multi-Criteria Decision-Making (MCDM) approach for NFRs trade-off was designed and implemented. The advantage of the present approach is that it first eliminates inappropriate infrastructures (VMs) based on hard constraints, such as CPU, memory, disk I/O which are necessary for service to run properly and for the purpose of providing NFR trade-offs only considers soft or desirable parameters which have the most influence on the application's QoS. The present approach was integrated into two recently designed and implemented cloud-based workbenches for the development and storage of container-based applications. The workbenches are not standalone components, and they rely heavily on the underlying SWITCH and ENTICE platform services, which allow dynamic container orchestration and storage in highly distributed environments. By employing the new MCDM approach in mainstream software engineering, the authors believe it would be possible to improve the QoS, shorten the software engineering time and save operational costs when developing smart IoT-based applications.

Acknowledgements

The projects have received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreements no. 815141 (DECENTER: Decentralised technologies for orchestrated Cloud-to-Edge intelligence) and no.643963 (SWITCH: Software Workbench for Interactive, Time Critical and Highly self-adaptive Cloud applications, and no. 644179 (ENTICE: Decentralised repositories for transparent and efficient Virtual Machine operations).

9 REFERENCES

[1] Sharp, J. H. & Ryan, S. D. (2010). Framework of Software Development Phases. *Data Base Adv. Inf. Syst., 41*(1), 56-75. https://doi.org/10.1145/1719051.1719055

[2] Baxley, K., Field, C., & De La Rosa, J. (2013). Deploying workloads with Juju and MAAS in Ubuntu 13.04.

[3] Fisher, S. (2014). An Introduction to Fabric8 and Why It's So Important for Integration. *DZone*.

[4] Shi, W. & Dustdar, S. (2016). The Promise of Edge Computing. *Computer (Long. Beach. Calif.), 49*(5), 78-81. https://doi.org/10.1109/MC.2016.145

[5] Fazio, M., Dustdar, S., Rana, O., Ranjan, R., & Villari, M. (2016). Osmotic Computing: A New Paradigm for Edge/Cloud Integration. *IEEE Cloud Comput., 3*(6), 76-83. https://doi.org/10.1109/MCC.2016.124

[6] Ganesan, M. & Sivakumar, N. (2017). A survey on IoT related patterns. *Int. J. Pure Appl. Math., 117*(19 Special Issue), 365-368.

[7] Hoßfeld, T., Schatz, R., Varela, M., & Timmerer, C. (2012). Challenges of QoE management for cloud applications. *IEEE Commun. Mag.*, 50(4), 28-36. https://doi.org/10.1109/MCOM.2012.6178831

[8] Baxley, K., De la Rosa, J., & Wenning, M. (2014). Deploying workloads with Juju and MAAS in Ubuntu 14.04 LTS.

[9] Fabric8, Fabric8 Documentation. Dec-2016.

[10] Garg, S. K., Versteeg, S., & Buyya, R. (2013). A framework for ranking of cloud computing services. *Futur. Gener. Comput. Syst.* 29(4), 1012-1023. https://doi.org/10.1016/j.future.2012.06.006

[11] Karim, R., Ding, C., & Miri, A. (2013). An end-to-end QoS mapping approach for cloud service selection. *Proc. - 2013 IEEE 9th World Congr. Serv. Serv, 2013*, 341-348. https://doi.org/10.1109/SERVICES.2013.71

[12] Wang, M., Meng, X., & Zhang, L. (2011). Consolidating virtual machines with dynamic bandwidth demand in data centers. *Proc. - IEEE INFOCOM*, (L), 71-75. https://doi.org/10.1109/INFCOM.2011.5935254

[13] Srikantaiah, S., Kansal, A., & Zhao, F. (2008). Energy aware consolidation for cloud computing. *Proc. 2008 Conf. Power aware Comput. Syst.*, 10.

[14] Zhang, L., Zhuang, Y., & Zhu, W. (2014). Constraint Programming based Virtual Cloud Resources Allocation Model. *Int. J. Hybrid Inf. Technol.*, 6(6), 333-344. https://doi.org/10.14257/ijhit.2013.6.6.30

[15] Singh, A., Korupolu, M., & Mohapatra, D. (2008). Server-storage virtualisation: Integration and load balancing in data centers. *2008 SC - Int. Conf. High Perform. Comput. Networking, Storage Anal. SC 2008*. https://doi.org/10.1109/SC.2008.5222625

[16] Ferdaus, M. H., Murshed, M., Calheiros, R. N., & Buyya, R. (2014). Virtual Machine Consolidation in Cloud Data Centers Using ACO Metaheuristic. (July), 306-317. https://doi.org/10.1007/978-3-319-09873-9_26

[17] Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., & Yuan, L. (2010). Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. *Proc. - 2010 IEEE 7th Int. Conf. Serv. Comput. SCC 2010*, 514-521. https://doi.org/10.1109/SCC.2010.69

[18] Durillo, J. J. & Prodan, R. (2014). Multi-objective workflow scheduling in Amazon EC2. *Cluster Comput., 17*(2), 169-189. https://doi.org/10.1007/s10586-013-0325-0

[19] Dong, J., Wang, H., & Cheng, S. (2015). Energy-performance tradeoffs in IaaS cloud with virtual machine scheduling. *China Commun.*, 12(2), 155-166. https://doi.org/10.1109/CC.2015.7084410

[20] Sun, Y., White, J., Eade, S., & Schmidt, D. C. (2016). ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimisation. *J. Syst. Softw.*, 116, 146-161. https://doi.org/10.1016/j.jss.2015.08.006

[21] Zhao, Z. et al. (2015). A software workbench for interactive, time critical and highly self-adaptive cloud applications (SWITCH). *Proc. - 2015 IEEE/ACM 15th Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2015*, 1181-1184.

[4] https://www.decenter-project.eu

https://doi.org/10.1109/CCGrid.2015.73
[22] Kimovski, D. et al. (2016). Towards an Environment for Efficient and Transparent Virtual Machine Operations: The ENTICE Approach. *Proc. - 2016 5th IEEE Int. Conf. Cloud Networking, CloudNet 2016*, 242-247. https://doi.org/10.1109/CloudNet.2016.30
[23] Štefanič, P. et al. (2019). SWITCH workbench: A novel approach for the development and deployment of time-critical microservice-based cloud-native applications. *Futur. Gener. Comput. Syst., 99*, 197-212. https://doi.org/10.1016/j.future.2019.04.008
[24] Štefanič, P. et al. (2018). Application-Infrastructure Co-Programming: managing the entire complex application lifecycle.
[25] Štefanič, P., Cigale, M., Jones, A. C., Knight, L., & Taylor, I. (2019). Support for full life cycle cloud-native application management : Dynamic TOSCA and SWITCH IDE. *Futur. Gener. Comput. Syst.* https://doi.org/10.1016/j.future.2019.07.027
[26] Wang, J. et al. (2017). Planning virtual infrastructures for time critical applications with multiple deadline constraints. *Futur. Gener. Comput. Syst.* https://doi.org/10.1016/j.future.2017.02.001
[27] Stankovski, V. et al. (2015). Towards an Environment Supporting Resilience, High-Availability, Reproducibility and Reliability for Cloud Applications. *Proc. - 2015 IEEE/ACM 8th Int. Conf. Util. Cloud Comput. UCC 2015*, 383-386.
[28] Trihinas, D., Pallis, G., & Dikaiakos, M. D. (2014). Jcatascopia: Monitoring elastically adaptive applications in the cloud. *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, 226-235. https://doi.org/10.1109/CCGrid.2014.41
[29] Metaheuristics, O., Durillo, J. J., Nebro, A. J., Luna, F., & Alba, E. (2006). Metal : a Java Framework for Developing Multi-Objective. *Sci. Technol., 01*, 1-12.
[30] Stefanic, P., Kimovski, D., Suciu, G., & Stankovski, V. (2018). Non-functional requirements optimisation for multi-Tier cloud applications: An early warning system case study. *2017 IEEE SmartWorld Ubiquitous Intell. Comput. Adv. Trust. Comput. Scalable Comput. Commun. Cloud Big Data Comput. Internet People Smart City Innov. SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2017*, 1-8. https://doi.org/10.1109/UIC-ATC.2017.8397637
[31] Kimovski, D., Saurabh, N., Stankovski, V., & Prodan, R. (2016). Multi-objective middleware for distributed VMI repositories in federated cloud environment. *Scalable Comput.*, 17(4), 299-312. https://doi.org/10.12694/scpe.v17i4.1202
[32] Casale, G. et al. (2016). Current and Future Challenges of Software Engineering for Services and Applications. *Procedia Comput. Sci., 97*, 34-42. https://doi.org/10.1016/j.procs.2016.08.278

Contact information:

Polona ŠTEFANIČ
Cardiff University, School of Computer Science and Information,
Queens Building 5, The Parade, Roath, Cardiff CF24 3AA, UK
E-mail: StefanicP@cardiff.ac.uk

Vlado STANKOVSKI
(Corresponding author)
University of Ljubljana,
Faculty of Computer and Information Science,
Večna pot 113, 1000 Ljubljana, Slovenia
E-mail: vlado.stankovski@fri.uni-lj.si